Ruby Classes

You will learn how classes are implemented in the Ruby language and how they are different.

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- ~~Scopes~~
- ~~Modules with hierarchies~~
- Inheritance hierarchy
- Ruby object model
- Method lookup
- Etc.

Insight 7:

Inheritance hierarchy

Ruby classes can inherit only from one class, but can also include as many modules as they want. Since ruby is a duck typed language, this works as multi-class inheritance.

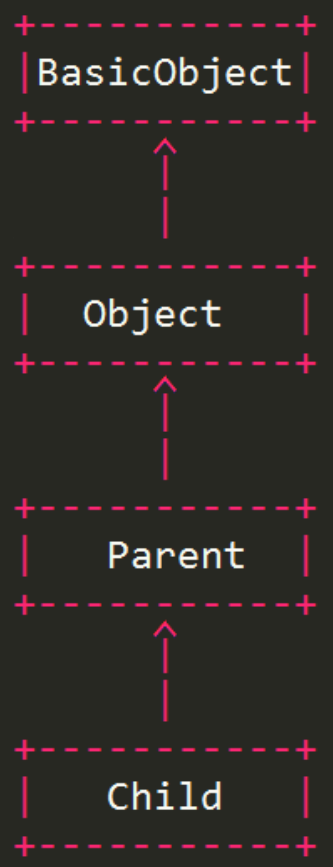
Class inheritance

```ruby
class Parent
end

class Child < Parent
end


Child.superclass                # Parent
Parent.superclass               # Object
Object.superclasss              # BasicObject
BasicObject.superclasss         # nil

String.superclass               # Object
Array.superclass                # Object
```



Class inheritance

- Single class inheritance, but you can include as many modules as you want
- All classes inherit from **Object**, which in turn inherits from **BasicObject**
- BasicObject was introduced in Ruby 1.9

Alternative way to get ancestors list

```ruby
Child.ancestors                    # [Child, Parent, Object, Kernel, BasicObject]


class Object
    include Kernel
    # ...
end
```

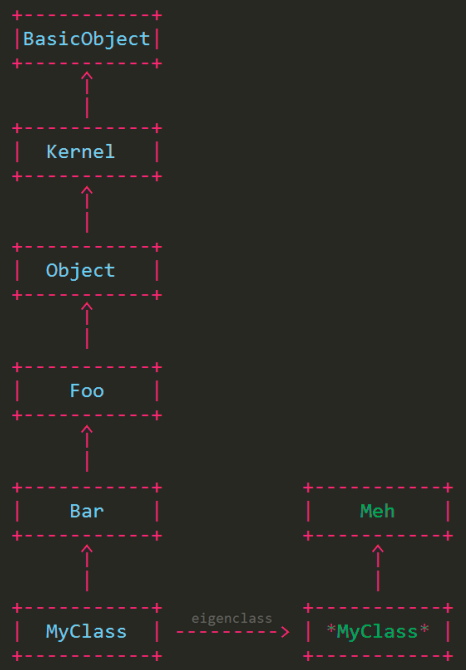
Modules in ancestry tree

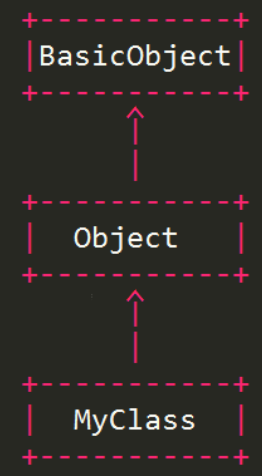
```ruby
module Foo
end

module Bar
end

module Meh
end

class MyClass
    include Foo
    include Bar
    extend  Meh
end
```






```ruby
MyClass.superclasss                 # Object
MyClass.ancestors                   # [MyClass, Bar, Foo, Object, Kernel, BasicObject]

mce = class << MyClass
        self
      end

mce.ancestors                       # [#<Class:MyClass>, Meh, #<Class:Object>, #<Class:BasicObject>, Class, Mc
```

Eigenclass hierarchy

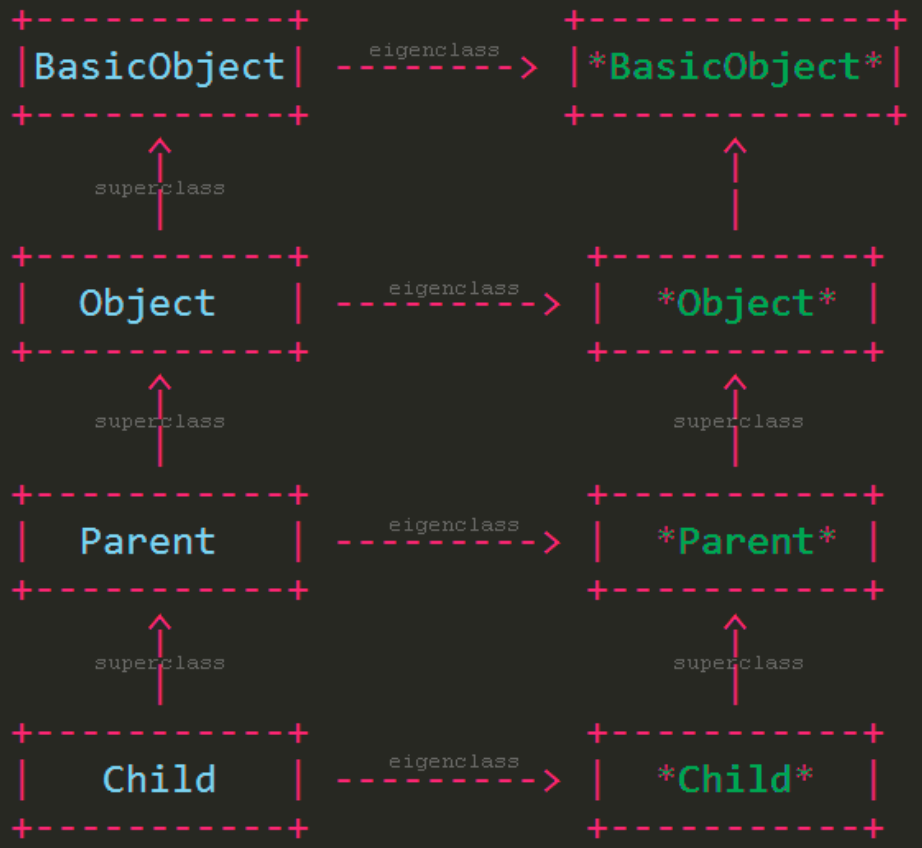
```ruby
class Parent
end

class Child < Parent
end

Child.superclass                # Parent
Parent.superclass               # Object
Object.superclasss              # BasicObject

che = class << Child
        self
      end                       # #<Class:Child>

sup = che.superclass            # #<Class:Parent>
sup = sup.superclass            # #<Class:Object>
sup = sup.superclass            # #<Class:BasicObject>
sup = sup.superclass            # Class
sup = sup.superclass            # Module
sup = sup.superclass            # Object
sup = sup.superclass            # BasicObject
sup = sup.superclass            # nil
```



Modules and class inheritance

- Modules included in the class are inserted into the class' inheritance tree
- Kernel module is included in the Object class
- Included modules are wrapped into the anonymous class and inserted into the class inheritance tree
- When you extend a class with a module, it is inserted into the class' eigenclass inheritance tree
- Eigenclass inheritance tree reflects class inheritance tree

Module ancestry

```ruby
module A
end

module B
    include A
end

B.ancestors                          # [:B, :A]
```

Our progress so far:

- ~~Open classes~~
- ~~Duck Typed~~
- ~~Just a runnable code~~
- ~~Object themselves~~*
- ~~Scopes~~
- ~~Modules with hierarchies~~
- ~~Inheritance hierarchy~~
- Ruby object model
- Method lookup
- Etc.